SYSTEMS AND METHODS FOR PRICING PRODUCTS

[0001] The present invention relates generally to a method of pricing products, and particularly to a method of reviewing product transactions in order to improve pricing strategies.

RELATED APPLICATIONS

[0002] This application is related to RULE-BASED SYSTEM FOR DETERMINING PRICE ADJUSTMENTS IN A PRODUCT CATALOG filed on April 12, 2002, attorney docket number 10953-006-999.

[0003] This application is related to SYSTEM AND METHOD FOR GROUPING PRODUCTS IN A CATALOG filed on April 12, 2002, attorney docket number 10953-005-999.

BACKGROUND OF THE INVENTION

[0004] Corporate optimization of pricing strategies is an important part of the successful operation of a business. In fact, pricing optimization is more important than a proportionate increase in volume. Consider, for example, the profit implications of a one percent increase in volume and a one percent increase in price. For a company with average economics, improving unit volume by one percent yields a 3.3 percent increase in operating profit, assuming no decrease in price. But, a one percent improvement in price, assuming no loss of volume, increases operating profit by 11.1 percent. See Marn and Rosiello, Harvard Business Review, September-October 1992, pp. 84-93. Improvements in price typically have three to four times the effect on profitability as proportionate increases in volume.

[0005] Because of the impact it has on profitability, substantial effort must constantly be undertaken to improve pricing. However, pricing issues are seldom simple and isolated. More typically, they are diverse, intricate, and linked to many disparate aspects of the business. While many companies track most pricing issues, a key aspect of price management, transaction price management, is often overlooked. Where concern at other price management levels is directed more toward the broad, strategic positioning of products in the markeplace, focus at the transaction level of price management is microscopic - customer by customer, transaction by transaction, deal by deal. Overlooking price management at the transaction level, the point where the product meets the consumer, has the potential to result in the loss of significant potential profits. For example, companies that use invoice price as a reporting measure overlook the difference between invoice and actual transaction price, which can mean significant reductions to bottom-line profit.

[0006] Companies that have identified the importance of transaction price management often use a pocket price waterfall to determine where their products' prices erode between list price and the actual transaction price. The price waterfall has a start price point and an end price point. Each of a series of price adjustments used to compute the end price point based on the start price point are represented in the waterfall as an element. The pocket price waterfall helps to address the critical issues of transaction price management. That is, the effective management of the price charged for each transaction. The waterfall details what base price is used and what terms, discounts, allowances, rebates, incentives, and bonuses apply.

[0007] An exemplary pocket price waterfall is illustrated in Fig. 1. The start price point for the exemplary waterfall is the dealer list price from which an order-size discount (quantity discount), customer class discount (customer discount), and payment terms (net 15) are subtracted and a freight charge is added to get the invoice price. The pricing adjustments order-size discount, customer class discount, payment terms, and freight charge each represent an element in the price waterfall. The magnitude of each element in the waterfall graph is typically the weighted average of the amount of the pricing adjustment that was applied to the product set covered by the waterfall during a predetermined time period. For example, the product set covered by Fig. 1 is shop keeping unit (SKU) number123 and the predetermined time period is the first quarter of 2002. The weighted average of the quantity discount applied to SKU number 123 during the first quarter of 2002 is ten dollars.

[0008] For companies that monitor price performance, the invoice price is the measure most commonly studied. See, e.g., Marn, *Id.* However, invoice price often does not reflect the true transaction amount. A host of pricing adjustments come into play between the invoice price and the final transaction cost. These pricing adjustments include prompt payment discounts, volume buying incentives, and cooperative advertising allowances. In the example shown in Figure 1, the pricing adjustments are a shop-keeping unit incentive and a volume incentive. When the income lost through these transaction-specific elements is subtracted from the invoice price, what is left is called the pocket price (final price), which is the revenues that are truly left in the company's pocket as a result of the transaction. For many companies, the pocket price, rather than the invoice price, is a better measure of the profitability of a particular transaction. In many instances, the amount of income lost through transaction-specific elements subtracted from the invoice price is substantial. For instance, the average decline from invoice price down to pocket price was 16.7% for one consumer packaged goods company, 17.7% for a commodity chemical company, 18.6% for a computer company, 20.3% for a footwear company, 21.9% for an automobile manufacturer, and 28.9% for one lighting products supplier. Marn, *Id.* A mere one percent improvement in these numbers would result in a substantial increase in the operating profits of any one of these companies.

[0009] Many companies do not focus on pocket price because accounting systems do not collect many of the off-invoice discounts on a customer or transaction basis. For example, payment term discounts often get buried in interest expense accounts, cooperative advertising is included in company wide promotions and advertising line items, and customer-specific freight gets lumped in with all the other business transportation expenses. Since these items are typically collected and accounted for on a company wide basis, it is often difficult to optimize pricing strategies at the transactional level. Yet, companies that do not actively manage the entire pocket price waterfall, with its multiple and highly variable revenue leaks, miss important opportunities to improve price performance.

[0010] In addition to merely providing price management down to the invoice level, rather than the transactional level, known price management systems are unsatisfactory because of their inherent complexity. An effective pricing strategy often requires the application of several different unique pricing adjustments to each product in the corporate catalog. A company that has a corporate catalog may have tens or hundreds of products in the corporate catalog. Thus, at any given time, an effective transactional level pricing strategy may require thousands of pricing adjustments. Further still, because of the complexity of the corporate pricing environment, which is affected by diverse variables such as supply cost, supply availability, inventory levels, competition, and seasonal demand, pricing adjustments must be changed over time in order to optimize overall pricing strategy. However, known price management systems do not provide a satisfactory interface for implementing both overall corporate strategy and the thousands of pricing adjustments often needed to effectively support this strategy.

SUMMARY OF THE INVENTION

[0011] In summary, the present invention addresses the drawbacks found in prior art pricing systems. The present invention provides a price management system that tracks corporate sales down to the transactional level and stores this transactional data in a novel database, termed the waterfall history database. This novel waterfall history database can then be used to produce detailed waterfall history graphs that allow for price management at all levels, including transactional price management. Another aspect of the present invention simplifies the execution of a pricing strategy by relegating most pricing adjustments into custom tables (price tables) that are free of complex pricing logic. No programing skills or knowledge of the overall corporate pricing strategies is needed to update these custom tables. Therefore, the level of skill needed to maintain the corporate pricing strategy is reduced, thereby lowering the labor costs needed to support an effective pricing strategy.

[0012] One embodiment of the present invention provides a method of graphically depicting a plurality of price adjustments that are applied to a product set over a predetermined time period. In the method, a request for a graphical depiction of the plurality of price adjustments that are applied to the product set over the predetermined time period is received. In one embodiment, the product set comprises a single shopkeeping unit (SKU) that uniquely represents a product in a catalog of products. In another embodiment, the product set comprises the set of products sold to a particular customer or a particular group of customers during the predetermined time period.

[0013] In response to the request, transaction data for the plurality of price adjustments for the product set during the predetermined time period are retrieved from a database that stores transaction data for the product. For each pricing adjustment in the plurality of price adjustments, a representation of an amount the price of the product set was adjusted in accordance with the corresponding price adjustment during the time period is computed using the transaction data. In some embodiments, each representation is a summation of the amount the price was adjusted by the corresponding price adjustment in each transaction for the purchase of the product set during the predetermined time period. In other embodiments, each representation is a weighted average of the amount the price was adjusted by the corresponding price adjustment in each transaction for the purchase of the product set during the predetermined time period. Each representation is graphed as an element in a graph having a start price point and an end price point. Each element is placed in the graph between the start price point and the end price point.

[0014] In some embodiments, the graph further includes one or more intermediate price points, such as an invoice price, and each element is associated with either an intermediate price point or the final price point. Further, each element is plotted in the graph before the price point associated with the element.

[0015] In still other embodiments of the present invention, a first price adjustment in the plurality of price adjustments includes a plurality of subcategories. For each subcategory of the first price adjustment, a representation of an amount the price of the product set was adjusted in accordance with the subcategory of the first price adjustment is computed for the time period using the transaction data. In such embodiments, the method further comprises accepting a selection of the element in the graph that corresponds to the first price adjustment. Then, for each subcategory of the first price adjustment, the representation that corresponds to the subcategory of the first price adjustment is graphed.

[0016] Another aspect of the present invention provides a method for computing the final price of one or more products in a price quote. In the method, a price trail is initialized with a starting point and an ending point. The ending point is used to compute the final price of the one or more products in the price quote. The price trail is populated with a plurality of price changes. Each price change illustrates a price change sequence that affects the ending point. At least one price change in the plurality of price changes is a compound price change sequence. A compound price change sequence includes a price change operator. A price changer operator performs an operation on one or more predetermined price change sequences. Once generated, the price trail is displayed and used to compute the final price of the one or more products in the price quote.

[0017] Another embodiment of the present invention is a method for computing a price for a product in a quote. A price form is provided for receiving the quote. The price form has a number of fields, including a field to specify the product, a field to specify a quantity of the product, and a field to specify a first agreement that governs the quote. The agreement includes a customer and an agreement coverage period. The price of the product in the quote is computed using a price function that corresponds to a designated field in the price form. The price function has an input that is taken from the designated field and an output that is used to determine the price of the product. The price function determines the output using the input value and a price table that includes a price adjustment value for the product. In some instances, the price table is a global price table. In other instances, the price table is agreement-specific and a price adjustment value in the price table is associated with a designated agreement. In such cases, when the designated agreement for the price adjustment value in the price table does not match the first agreement, the price adjustment value is not used by the price function to determine the price, and, when the designated agreement for the price adjustment value in the price table matches the first agreement and the price adjustment value is for the product, the price adjustment value is used by the price function to determine the price.

[0018] In some embodiments, each price adjustment value in the price table has a priority. When two or more price adjustment values in the price table apply to the product, the price adjustment value having the highest priority is used by the price function to determine the price for the product. In still other embodiments, the price adjustment value in the price table is associated with a product, a product collection, or a product category.

[0019] In some embodiments, the method for computing a price for a product in a quote further comprises initializing a price trail with a starting point and an ending point. The starting point corresponds to the list price for the product and the ending point corresponds to the final price for the product in the price quote. The price trail is populated with a plurality of price changes. Each price change in the plurality of price change affects the value of the ending point in the price trail. Finally, the price trail is displayed.

[0020] Another aspect of the present invention provides a computer program product for use in conjunction with a computer system. The computer program product comprises a computer readable storage medium and a computer program mechanism embedded therein. The computer program mechanism comprises a waterfall reporting module for graphically depicting a plurality of price adjustments that are applied to a product set over a predetermined time period. The waterfall reporting module includes instructions for receiving a request for a graphical depiction of the plurality of price adjustments that are applied to the product set over the predetermined time period. The module further includes instructions for retrieving, in response to the request, transaction data for the plurality of price adjustments for the product set during the predetermined time period from a waterfall history database that stores transaction data for the product. The module also includes instructions for computing, for each price adjustment in the plurality of price adjustments, a representation of an amount the price of the product set was adjusted in accordance with the corresponding price adjustment during the time period using the transaction data. Finally, the module includes instructions for graphing each representation as an element in a graph having a start price point and an end price point. In this graph, each element is placed between the start price point and the end price point.

[0021] Yet another aspect of the present invention provides a computer program product for use in conjunction with a computer system. The computer program product comprises a computer readable storage medium and a computer program mechanism embedded therein. The computer program mechanism comprises a price function. The price function comprises instructions for initializing a price trail with a starting point and an ending point. The ending point corresponds to a final price for the product set in the price quote. The price function further includes instructions for populating the price trail with a plurality of price changes. Each price change in the plurality of price changes illustrates a price change sequence that affects the final price of the product set in the price quote. Further, a price change in the plurality of price changes is a compound price change sequence that includes a price change operator that performs an operation on one or more predetermined price change sequences affecting the final price of the product set. The price function also includes instructions for displaying the price trail and instructions for computing the price trail based on the plurality of price changes.

[0022] Still another aspect of the present invention provides a computer program product for use in conjunction with a computer system. The computer program product comprises a computer readable storage medium and a computer program mechanism embedded therein. The computer program mechanism comprises a price engine for computing a price for a product in a quote. The price engine includes a price form for receiving the quote. The price form includes a field to specify the product, a field to specify a quantity of the product, and a field to specify a first agreement that governs the quote. The agreement includes a customer and an agreement coverage period. The price engine further includes a price function that corresponds to a designated field in the price form. The price form is used to compute the price of the product in the quote. The price function has an input that is taken from the designated field and an output that is used to determine the price for the product. The price function includes instructions for determining the output to the function based upon the input value and a price table. The price table includes a price adjustment value for the product.

[0023] Another aspect of the invention provides a computer system for graphically depicting a plurality of price adjustments that are applied to a product set over a predetermined time period. The computer system includes a central processing unit and a memory, coupled to the central processing unit. The memory stores a waterfall history database and a waterfall reporting module. The waterfall history database stores transaction data for the product. The waterfall reporting module includes instructions for receiving a request for a graphical depiction of the plurality of price adjustments that are applied to the product set over the predetermined time period. The module further includes instructions for retrieving, in response to the request, transaction data for the plurality of price adjustments for the product set during the predetermined time period from the waterfall history database. The module further includes instructions for computing, for each price adjustment in the plurality of price adjustments, a representation of an amount the price of the product set was adjusted in accordance with the corresponding price adjustment during the time period using the transaction data. Finally, the module includes instructions for graphing each of the representations as an element in a graph having a start price point and an end price point. Each element is placed between the start price point and the end price point in the graph.

[0024] Still another aspect of the present invention provides a computer system for computing the final price of one or more products in a price quote. The computer system comprises a central processing unit and a memory, coupled to the central processing unit. The memory stores a price function and a price trail. The price function comprises instructions for initializing the price trail with a starting point and an ending point. The ending point corresponds to a final price for the product set in the price quote. The price function further comprises instructions for populating the price trail with a plurality of price changes. Each price change in the plurality of price changes illustrates a price change sequence that affects the final price of the product set in the price quote. At least one price change in the plurality of price changes is a compound price change sequence. The compound price change sequence includes a price change operator that performs an operation on one or more predetermined price change sequences affecting the final price of said product set. Finally, the price function includes instructions for displaying the price trail and instructions for computing the price trail based on the plurality of price changes in the price trail.

[0025] An additional aspect of the present invention provides a computer system for computing a price for a product in a quote. The computer system includes a central processing unit and a memory that is coupled to the central processing unit. The memory stores a price engine and a price table. The price table includes a price adjustment value for the product. The price engine includes a price form and a price function. The price form receives the quote. The price form includes a field to specify the product, a field to specify a quantity of the product, and a field to specify a first agreement that governs the quote. The agreement includes a customer and an agreement coverage period. The price function corresponds to a designated field in the price form. The price form computes the price of the product in the quote. The price function has an input that is taken from the designated field and an output that is used to determine the price for the product. The price function includes instructions for determining the output to the function using the input value and the price table.

BRIEF DESCRIPTION OF THE DRAWINGS

[0026] Additional objects and features of the invention will be more readily apparent from the following detailed description and appended claims when taken in conjunction with the drawings, in which:

[0027] Fig. 1 illustrates a waterfall graph in accordance with the prior art.

[0028] Fig. 2 illustrates a computer system for processing quotes for products and for optimizing corporate pricing strategy in accordance with one embodiment of the present invention.

[0029] Fig. 3 illustrates a price form in accordance with one embodiment of the present invention.

[0030] Fig. 4 illustrates the processing steps used to determine the price of a product using the data structures and program modules found in one embodiment of the present invention.

[0031] Fig. 5 illustrates the architecture of a price trail data structure 291 in accordance with one embodiment of the present invention.

[0032] Fig. 6 illustrates a price trail in accordance with one embodiment of the present invention.

[0033] Fig. 7 illustrates a price trail having a compound price change in accordance with one embodiment of the present invention.

[0034] Fig. 8 illustrates the display and computation of a price trail in accordance with one embodiment of the present invention.

[0035] Fig. 9 illustrates an improved price waterfall graph in accordance with one embodiment of the present invention.

[0036] Fig. 10 illustrates a price agreement in accordance with one embodiment of the present invention.

[0037] Like reference numerals refer to corresponding parts throughout the several views of the drawings.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0038] Fig. 2 illustrates a system 210 that is operated in accordance with one embodiment of the present invention. System 210 includes at least one computer 220 that is optionally connected to one or more other computers by a communications network (not shown). The communications network used to connect computer 220 to other computers is a local area network (LAN), wide area network (WAN), metropolitan area network (MAN), an Intranet, the Internet, or any combination of such networks.

[0039] Computer 220 includes standard components including a central processing unit 222, memory 224 (including high speed random access memory) for storing program modules and data structures, user input/output device 226, a network interface 228 for coupling computer 220 to other computers via a communication network (not shown), and one or more busses 234 that interconnect these components. User input/output device 226 includes one or more user input/output components such as a mouse 236, display 238, and keyboard 208.

[0040] Memory 224 includes a number of modules and data structures that are used in accordance with the present invention. It will be appreciated that at any one time during operation of system 200, a portion of the modules and/or data structures shown in Fig. 2 as being stored in memory 224 will be stored in random access memory while another portion of the modules and/or data structures will be stored in non-volatile storage 242. Non-volatile storage 242 is typically accessed using disk controller 240.

[0041] Memory 224 includes one or more price engines 250 for computing a price for a product in a quote. In one embodiment, a price engine 250 is a single script that is used to encode a corporate pricing strategy. Memory 224 may include more than one price engine 250, however, in one embodiment of the present invention, only one price engine is active at any given time. A price engine defines the price trail data structures 291, price forms 252, price tables 254, and price models 256 that are used to define a corporate pricing strategy.

PRICE FORMS 252

[0042] A price engine 250 references one or more price forms 252 that are stored in memory 220. Each price form 252 provides a mechanism through which data is provided to price engine 250. In one embodiment, this mechanism is a field to specify the product for which a quote is desired, a field to specify a quantity of the product, and a field to specify a price agreement 288 (Fig. 2) that governs the quote. In some embodiments, each price agreement 288 includes a customer identifier, an agreement coverage period, and the identity of one or more products or services in product catalog 294 that are covered by the agreement. For example, an exemplary price agreement 294 may designate a customer "FooBar" for the first quarter of 2002. The exemplary price agreement 294 may further designate that the products covered by the agreement are one or more specified products, such as SKU 123 and SKU 124. In addition, the exemplary price agreement 294 may designated one or more product collections or product categories that are covered by the price agreement. Product collections and product categories are discussed in further detail below.

[0043] A price form 252 can be displayed on a screen 238 (Fig. 2) for data entry. Alternatively, a price form can be used by an external system (not shown) to submit data to price engine 250 for pricing. Fig. 3 illustrates a price form 252 in accordance with one embodiment of the present invention. The exemplary price form 252 of Fig. 3 consists of three areas, a header 302, a table of line items containing line fields 304, and a footer 306. Header 302 displays information about the form such as the account 308 to which it belongs, and the name 310 of the individual requesting the quote. The line fields 312 in the table of line items 304 are displayed as rows within table 304. Typical fields displayed in the line fields 312 are product description, quantity, unit price, and extended price. In one embodiment, footer 306 is used to apply form-level charges such as, for example, the order total, freight charge, a sales tax charge, and the order total charge.

[0044] In some embodiments of the present invention, price forms 252 have control buttons for performing actions on the form. For example, exemplary price form 252 in Fig. 3 includes an "add products" button 320 for opening a catalog in a new window so that user 310 can add products to the order, a "price" button 322 that is used to recalculate the prices on price form 252 when the quantity ordered has been changed, and a "save" button that saves the information entered into price form 252. Additional exemplary buttons include, but are not limited to, a "suggest" button for providing a suggestion for how the customer might get a better price and a "projected price" button for calculating the net price for the order after any off-invoice programs are applied.

[0045] In one embodiment of the present invention, a specialized programming language is used to define a price form 250. In this programming language, each field in a line field 312 has a data type, such as integer, long, float, double, money, percentage, data, bigdecimal, Boolean, adjustment, or string. In addition, each field may have a value expression that specifies a formula to calculate the field value or the location where the value for the field is located. Further, each field in line field 312 may have a marker that indicates that the field value is to be calculated using a price model 256 (Fig. 2). Price models 256 are described in more detail below. If a field does not have a value expression or a marker, then it is designated an input-only field whose value is entered by a user. An exemplary script, written in the specialized programming language that is used to generate the form 250 illustrated in Fig. 3, has the format:

```
101  pricingform Quote {
102  description "Basic Quote Form" ;

103  headerfield QuoteNumber        { type integer; label "Number" ; noneditable ;  }
104  headerfield Account            { type Account; }
105  headerfield Agreement          { type PriceAgreement; }
106  headerfield ValidTo            { type date; label "Valid Until" ; }
107  headerfield ShippingMethod     { type ShippingMethod; }
108  headerfield Description        { type string(1024); label "Quote Description" ; }

109  linefield Qty        { type integer; minimum 1; }
110  linefield Product    { type Product; }
112  linefield ListPrice { type money; value Product.ListPrice; label "List Price";
                                                            noneditable;}
113   linefield SalesPrice { type money; vslcalculated ; noneditable ; vptmultiplier Qty ; }
114   linefield LineTotal      { type money; value SalesPrice * Qty; noneditable; }

115   footerfield SubTotal             { type money; value SUM(LineTotal); noneditable; }
116   footerfield SubtotalAdjustments        { type money; vslcalculated ; noneditable ; }
117   footerfield Total                      { type money;  noneditable ; value SubTotal +
                                                            SubtotalAdjustments; } }
```

In the exemplary code, lines 101 and 102 define a new pricing form. Lines 103 through 108 define header 302, lines 109 through 114 define table 304, and lines 115 through 117 define footer 306.

PRICE TABLES 254

[0046] Each price engine includes (i.e. references) one or more price tables 254 (Fig. 2). Price tables 254 include auxiliary information used to determine pricing, such as price adjustments for one or more products offered by a vendor. Prices tables 254 of the present invention are advantageous because they allow for the maintenance of price adjustments without the use of complex programming statements. For example, a simple price table in accordance with the present invention contains a column for product and a column for a corresponding price or adjustment.

[0047] Each price table 254 can be either a global price table or an agreement-specific price table. Global price tables contain data that applies regardless of the agreement in effect for a particular form. Agreement-specific price tables contain information that applies only when a specific price agreement 288 (Fig. 2) is in force. Agreements 288 are discussed in further detail below. One embodiment of the present invention provides a price table 254 that is agreement-specific. In this embodiment, a price adjustment value in the price table 254 is associated with a designated price agreement 288. Further, when the designated price agreement 288 for the price adjustment value in the price table 254 does not match the price agreement governing a price quote, the price adjustment value is not used by the price function 270 to determine the output to the function. Likewise, when the designated agreement 288 for the price adjustment value in the price table 254 matches the price agreement 288 governing the price quote and the price adjustment value is for the product designated in the price qutoe, the price adjustment value is used by the price function 270 to compute the price function 270 output value.

[0048] In many enterprises, the product catalog 294 (Fig. 2) is extremely large. In addition, enterprises often consider many factors when setting discounts. Each such factor may require a column in a price table 254. Consequently, price tables 254 can become large. In order to handle these large price tables 254, the present invention provides a number of different methods for simplifying the maintenance of the tables. One technique is to define product groups that contain many products and then to assign a price adjustment to the entire group in just a single row of a price table 254. The present invention provides two different product groups. They are product categories and product collections. A product category is a defined subset of products in a product catalog 294. A product collection is generated by making a query of product catalog 294 using some form of query. The query may be static in the sense that the query is not a function of any environmental variable. Alternatively, the query may be arbitrary in the sense that the query is dynamically generated. Product collections formed using dynamically generated queries are referred to as dynamic collections. An example of an arbitrary query that is dynamically generated is a request for any product in product catalog 294 in which there are more than 500 units of the product in the inventory. Within price tables 254, any entry that can contain products may also contain product groups. Any adjustment amount listed for that product group will be applied to all products in the group. Accordingly, in one embodiment of the present invention, each price adjustment value in a price table 254 is associated with a product, a product collection, or a products category.

[0049] Another mechanism that can be used to manage large price tables 254 is the use of a priority column. To illustrate, an exemplary row (Fig. 3) in a price table 252 may have the columns "product", "discount", and "priority", where each entry in the "product" column identifies a particular product in product database 294 or a product collection or a product category, each entry in the "discount" column is a price adjustment that applies to the product, and each entry in the "priority" is a number that determines the priority of the row relative to all other rows within pricing table 254 that are applicable to the designated product. So, for example, in the case where a given pricing table 254 has two rows that apply to the same product, the row that includes the highest priority will be used by pricing engine 250 to adjust the price of a product. The use of priority can also be used to resolve rows that include product groups. To illustrate, consider the following price table 254:

Table 1: Exemplary Price Table 254

Product	Discount	Priority
120Series	4%	5
#127	3%	10
...	...	...

In this pricing table, product #127 is a member of group "120Series" and should receive a four percent discount. The product is also listed independently on a product row. Because the priority on the product row is higher, the system will apply a three percent discount for SKU #127, even though the product belongs to the group 120Series. In one embodiment of the present invention, in the case where two rows have the same priority, pricing engine 250 will apply both adjustments. For example, in Table 1, if both rows have a priority of five, then a customer would get a total of a 7% discount on product #127. In other embodiments, if two or more rows have the same priority five, pricing engine 250 will apply the largest adjustment, the smallest adjustment, or the adjustment resulting in the lowest price.

[0050] Some aspects of the present invention use pricing agreements. In general, a pricing agreement defines how products are priced under that agreement. A customer may have many accounts, each of which can have many price agreements. In some embodiments, a pricing agreement includes standard information like customer name and validity dates, a price recipe that determines the pricing strategy for the agreement, and a set of pricing tables that are filled with content that is appropriate for the agreement.

[0051] Pricing agreements are often similar. In such situations, the use of inheritance is a beneficial mechanism for managing price tables 254. For example, some larger customer organizations may have many accounts that are covered by a particular pricing engine 250. The customer organization may negotiate an umbrella price agreement 288 that covers pricing for all accounts. Each individual organizational unit may wish to either extend the umbrella agreement 288 to cover additional products or override agreed upon prices for some products or groups (product collections and/or product categories). The inheritance mechanism permits this form of negotiation. When defining a customer agreement 288, the user specifies that the agreement is a "child" agreement. When an agreement is designated in this manner, all price tables defined for the umbrella agreement 288 for the customer agreement are copied. Each of the child agreements inherit the price tables defined for the umbrella agreement. If the umbrella agreement is subsequently updated, the updates will automatically propagate to each child agreement. Within the price tables of a child agreement, user can add rows for products or groups not covered in the umbrella agreement. The user may also add new rows for product groups and groups covered by the umbrella agreement and assign them a different adjustment value with a higher priority in order to override the inherited adjustment values.

[0052] Still another mechanism for managing price tables is the use of vectors. Vectors are a technique a table designer can use to reduce the number of rows needed in a price table. Vectors are useful in situations such as the case where the size of an applicable price adjustment is a function of the customer's credit rating. Without the use of vectors, a single product would require **n** rows in a price table 250, where **n** is the number of different credit ratings used in the applicable credit rating system. A vector is defined using a separate table. In the credit rating example, the vector is a 2 by **n** table, which specifies **n** credit categories (e.g., fair, good, excellent) and the respective price adjustment for each of these categories (e.g. -2%, -5%, -8%). Thus, when the vector is used, the price table no longer needs to display credit rating and the respective adjustment because these are determined by the vector. The table now contains only one row for each product (or product group). When the user adds a new row, or selects an existing row, a frame is displayed below the table. When a new row is created, a new vector is created for that row. The vector contains one row for each possible value of credit rating. The user simply has to type in the adjustment amounts desired.

[0053] A commonly used type of vector is called a volume break. Table 2 shows a volume break vector for one product:

Table 2: Exemplary volume break vector

1-99	1%
100-999	2%
1,000-10,000	3%

A volume break vector contains two columns. In the first column, a user specifies an integer range. In the second, the user specifies an adjustment amount. A volume break is useful in cases where a discount based on the number of units purchased is needed.

[0054] Another table management tool is a matrix. A matrix differs from a vector only in the sense that a matrix contains three or more columns instead of two. Table 3 illustrates an exemplary matrix in which the price adjustment for a product is defined for each possible combination of two variables.

Table 3: Exemplary Matrix

Gold	Fair	-2%
Gold	Good	-5%
Gold	Excellent	-8%
Silver	Fair	-1%
Silver	Good	-3%
Silver	Excellent	-5%
Bronze	Fair	-1%

Bronze	Good	-2%
Bronze	Excellent	-3%

[0055] In Table 3, there are three possible values for customer class and three possible values for credit rating for a total of nine combinations. There is one row in the matrix for each combination in order to specify a unique adjustment amount for each combination. Without the use of a matrix, the price table would need to contain nine rows for each product (or product group). Using the price matrix illustrated in Fig. 3, each product or product group only requires a single line. Matrices and vectors can be combined with priority overrides, product groups, and inheritance to produce concise and powerful price tables.

[0056] Vectors, volume breaks, and matrices are linked to prices tables 254 using a two step process. In one step, the vector, volume break, or matrix is defined. For example, the matrix "ClassVsRating" may defined as:

```
matrix ClassVsRating {
        field custclass {type CustomerClass; rowindex;}
        field rating     {type CreditRating; columindex;}
        field discount  {type adjustment; }
}
```

In the other step, the price table is defined. The vector, volume break, or matrix is uses as the data type for one column in the price table 254. For example, the price table 254 "ClassVsRatingDiscount" may have the form:

```
globaltable ClassVsRatingDiscount {
        column Product        {type Product; expansionshortcuts;}
        column Discount       {type ClassVsRating;}
}
```

In table "ClassVsRatingDiscount", the column "Discount" will make use of the vector "ClassVsRating".

[0057] In some embodiments, price tables are defined using a special form of programming lagnauge. Once defined, no programming statements or programing knowledge is required to maintain the price tables. Price tables can be either global or agreement-specific. In some embodiments, when a global price table is created, a priority column will automatically be added to the table. In some embodiments, when an agreement-specific table is created, price engine 250 will automatically create an agreement column and a priority column. The agreement column is used to specify which price agreement 288 is associated with each row of data in the agreement-specific table. An agreement-specific table may have some rows that are associated with one price agreement 288 and some rows that are associated with a different prices agreement 288. In some embodiments, a price function 270 is activated by a price form to search an agreement-specific table. Further, the price form uses a particular price agreement 288. In such instances, the price function 270 ignores rows in the agreement-specific table that are not associated with the price agreement 288 used by the price form.

[0058] In some embodiments of the present invention, each column in a price table 254 has a data type that constrains the type of information that can be stored in that column. For example, the column "Quantity" would typically have a data type of integer. This prevents the storage of different data types, such as string or a Boolean value, in the field. In addition, a number of other constraints may be specified for each a column of a price table 254.

[0059] In some embodiments of the present invention, a price table 254 will have at least one column that acts as a search key. The search key is the column used in price functions 270 (Fig. 1) to search for specific rows within price table 254. For example, a price function 270 may have a variable whose value identifies a particular product. The prices function 270 may search a price table 254 consisting of a product and a discount column. The price function 270 will search the product column of the price table 254 to find a row containing the identified product. In this case, the product column is the search key.

[0060] In typical embodiments, every price table 254 contains one or more value columns. Value columns contain the information that a price function 270 needs to look up. In the example provided above, the discount column is the value column. A price function 270 performs a search that looks something like this pseudo-code expression:

```
if Table.Product = MyProduct THEN Adjust -Table.Discount
```

Although product is a common search key, other forms of search keys are possible. For example, a price table 254 that has customer as the search key column and discount as the value column is possible. Price table 254 may include any number of columns and may use multiple columns as the search key.

[0061] Common price table constraints that may be added to price table 254 include "unique", "required", "index" and "expansion/shortcut". The "unique" column property specifies that, for every row in the column, any value entered into the column must be unique. This constraint is useful in situations where the data type for the each field in the column has been set to string. In one embodiment, when a user creates a new row and enters a non-unique value in a column having the "unique constraint", the row will not be saved into the table. The constraint "required" is used for a column to specify that every field in the column must have a value. The constraint "index" is used to indicate that the column will be used by an applicable price function 270 to search the table. Declaring that a column is an "index" allows for user controlled search optimization, causes the system to generate an index for the column that facilitates fast searches, and consequently improves search performance times. The constraint "expansion/shortcut" is typically used for a column having the data type Product. When expansion shortcuts are introduced into such a column, the user has the ability to enter product groups as the value of this column as well as individual products. Thus, the "expansion/shortcut" constraint regulates whether or not a given column in a price table 254 can be used to enter product groups such as product collections and product categories.

[0062] In some embodiments of the present invention, each price table 254 has a special keyword (*e.g.,* "agreementtable"or "globaltable") followed by a name for the table. One such keyword (*e.g.,* "agreementtable") defines the price table 254 as an agreement table. Another such keyword (*e.g.,* "globaltable") defines the price table 254 as a global table.

[0063] Each price table 254 has two or more columns. Each column has a name and a set of column attributes. Further, each column optionally has a description and/or one or more properties. Column attributes include a data type. Optionally, each column has a specification of minimum and maximum values allowed in the column, a label that is used as the column name when displaying the table in the user interface, and an "expansionshortcuts" flag to indicate whether short-cuts are allowed in the column.

An exemplary script, written in the specialized programming language that is used to generate a price table 254 has the format:

```
201 agreementtable FixedPriceTable {
202    column product          { type Product; expansionshortcuts; property index }
203    column price            { type money; } }

204 globaltable SKUPromotions {
205    description " Promotions for specific SKUs";
206    column SKU            { type Product; property index }
207    column ValidFrom      { type date; }
208    column ValidTo        { type date; }
209    column Discount       { type adjustment; minimum 0; }}
```

Lines 201 through 203 of the exemplary script define an agreement table 254 having the name "FixedPriceTable". Lines 202 and 203 respectively define a product column having type "product" and a price column having type "money". Lines 204 through 209 define a global price table 254 having the name "SKUPromotions". Line 205 provides a brief description of the table while lines 206 through 209 define the properties of four columns found in the price table.

[0064] Price tables 254 are advantageous because they provide a satisfactory interface for implementing the thousands of pricing adjustments often needed to effectively support a corporate pricing strategy. No programing skills are needed to populate or maintain price tables 254 and the present invention provides numerous methods, described above, for maintaining potentially large classes of price adjustments found within such tables.

PRICE MODELS 256

[0065] Price models specify how prices are calculated for each field in a price form 252 that has a marker that indicates that the field value is to be calculated using a price model

256. Price models can be used broadly across a wide range of pricing situations, especially if their price functions 270 have been parameterized using look-ups from price tables 254. Thus, a typical company will have only a handful of pricing strategies that are so sophisticated they require their own price models 256. Most price strategies can be implemented using one price model 256 by applying different data values from price tables 254. This design flexibility means that price models 256 are relatively stable.

[0066] In one embodiment of the present invention, a price model 256 consists of a price name 258, a specification 260 of the price trail data structure 291 that the price model 256 uses, specifications 262 of the price forms 252 for which the price model provides price functions 270, and a price function 270 for each field in a supported price form 252 that has a marker indicating that the field value is to be calculated by the price model 256. To illustrate, exemplary code for the price model "direct sales" is provided:

```
301 pricemodel DirectSalesPriceModel {
302       forwaterfall GlobalWaterFall;

303   usesform Quote {

304     Code for SalesPrice input Product.ListPrice output SalesPrice  {
305           /* Code for calculating the sales price field of the quote form comes here.
306            * The input value for the Code is the product list price, and the output
307            * will be something called SalesPrice, which the code will specify here. */}

308     Code for SubtotalAdjustments input SubTotal output FinalTotal - SubTotal {
309           /* Code for calculating the sub-total adjustment field of the quote form comes
310            * here.  The input value for the code is the SubTotal field, and the output
311            * will be FinalTotal - SubTotal, which the code will specify here. */}}
```

In this exemplary code, line 301 defines a price model 256 entitled "DirectSalesPriceModel" that stores transactional data to the price trail data structure 291 entitled "GlobalWaterFall". Price trail data structures 291 are defined in more detail below. Line 303 states that the model uses the price form 252 "Quote". In code not shown, the price form "Quote" is defined to include two fields having a marker indicating that the field value corresponding to the markers is calculated by a particular price function in the model entitled "DirectSalesPriceModel". In particular, price form "Quote" has a field value that is computed by the function "SalesPrice", which is defined by lines 304 through 307 of the exemplary code. In addition, the price form "Quote" has a field value that is computed by the function "SubtotalAdjustments", which is defined by lines 308 through 308 through 311 of the exemplary code. Details of the functions "SalesPrice" and "SubtotalAdjustments" are not shown in order to highlight the relationship between fields with markers in price tables 252, and the price functions 270 that compute the values for such fields that are found in price models 256. More details on price functions 270 is provided below.

PRICE FUNCTIONS 270

[0067] As outlined above, price functions 270 correspond to designated fields in price forms 252. Price forms 252 are used by price models 256 to compute the price of a product in a quote. In some embodiments, each price function 270 has an input that is taken from the designated field and an output that is used to determine the price for the given product. Price functions 270 include instructions for calculating the output using the input value and one or more price tables 254.

[0068] In typical embodiments, there is one price function for each designated (marked) field in a price form 252. In some embodiments, each price function 270 has an input value 272 of data type "money" and an output value 274 of data type "money." That is, all price functions in accordance with such embodiments of the present invention accept one input value 272 and return one output value 274. The input value 272 passed to a price function 270 is used as the initial price value that the price function 270 uses. The output value of the function can be one of two things: (i), the value of the final price (or other price) for the product or product set or (ii), the difference between the input value 272 value and some adjustment amount. In either case, the output 274 is used to determine the final price for a product or product quote in accordance with a form 252 used by a given price model 256 in a given price engine 250. In some cases, the one or more products in a price quote comprise a single shopkeeping unit (SKU) that uniquely represents a single product in product catalog 294. In other cases, the one or more products in a price quote comprise a plurality of products in the product catalog.

[0069] In some embodiments, a price function 270 has a function keyword followed by an identifier 276. Identifier 276 represents the name of the field in a price form 252 for which this function is being defined. Further, the price function 270 in accordance with this embodiment includes an input keyword followed by an expression. The expression represents the input value 272 to the price function. The expression may be an alias defined elsewhere in a price engine 250. The expression will represent a field defined on the corresponding price form 252. Often the field is the field that contains the list price for a product. In addition, the exemplary price function 270 includes an output keyword followed by an identifier. The identifier represents a variable (output value 274) defined within the body of the price function 270. Optionally, a subtraction symbol followed by a second identifier may be specified. In one embodiment of the present invention, a price function 270 may be specified using the syntax:

```
VSL = 'vsl' 'for' ident 'input' Expr 'output' ident [ '-' ident ] [ 'vptmultiplier' ident ]
```

[0070] To illustrate the utility of price adjustments, the following example is provided. In this example, a price function 270 is used to calculate an adjustment to a price rather than the final price. It will be appreciated from this example that functions 270 may similarly be used to compute the final price for one or more products in a price quote.

```
// Calculate the amount for an adjustment field within price form footer
// Use Form.Subtotal as input
// Return the difference between FinalTotal and SubTotal, as output

// A price function output may be a subtraction expression, not just a variable
  code  for SubtotalAdjustments input SubTotal output FinalTotal - InitialTotal {

// Define the first price value (is equal to input)
pricevalue InitialTotal;

//Calculate the adjustment amount
if (SubTotal > 10000) then

// Allocate the adjustment amount to each line item by percentage of
// Form.LineItem.QuotePrice divided by Form.LineTotal
// write the value of the adjust amount to the indicated category in reporting
// database

adjust "Big Order Discount" - 5 % allocation prorata LineTotal;
end ;

//FinalTotal = SubTotal - Adjustment Amount
```

```
417    pricevalue FinalTotal;
```

In lines 401 through 405 of the illustrative pseudocode, the field Form.Subtotal from the price form 252 having the name "Form" is taken as the input value 272. The price function 270 will use the input value to calculate an output value 274 which is the difference between FinalTotal and SubTotal. In lines 408 through 415, the value "Adjustment Amount" is computed based on the initial value of SubTotal (line 409). Further, the adjustment amount is reported to a price trail data structure 291 in a waterfall history database (lines 412-413). In line 416, the output value 274 "FinalTotal"is computed as the difference between "SubTotal" and "Adjustment Amount". In line 416, the output value 274 is reported to the designated field in the calling price form 252.

[0071] In some embodiments of the present invention, price trail data structures 291 are referred to as price trails and each price function 270 generates a price trail each time the price function 270 is executed. In some embodiments, a price function 270 includes instructions for initializing a price trail with a starting point and an ending point. The ending point may be a final price for the product or product set in the price quote. Alternatively, the ending point of the price trail may be some intermediate price point used in the quote generation process, such as the invoice price. The ending point of the price trail is the value returned by function 270. The starting point, the ending point, and any intermediate price point between the starting point and the ending point each comprise totals or subtotals within the price calculation that is performed by a price function 270. Each price trail starts with and ends with a price value.

[0072] In some embodiments, a pricing function 270 includes instructions for populating the price trail with a plurality of price changes. Each price change in the plurality of price changes affects a value of the price trail ending point. Fig. 6 shows an exemplary price trail 600. The construct "List Price" 604 is the starting point and "LineItem Price" 610 is the ending point of price trail 600. The two price changes in price trail 600, represented by circles, are a two percent reduction for the node 606 "Customer class" and a two dollar reduction for the node 608 "Year end promotion".

[0073] The price trail of Fig. 6 is a price trail in which each price change is simple. That is, each price change does not operate on other price changes. In some instances, however, a price trail will include a price change that is a compound price change. A compound price change includes a price change operator that performs an operation on one or more predetermined price changes. Fig. 7 illustrates a price trail 700 that includes the compound price change "MAX" (708). Like price trail 600, the construct "List Price" 704 is the starting point and "LineItem Price " 710 is the ending point of price trail 700, and the price change "Customer class" 706 is a simple price change. The complex price change 708 takes the maximum of two predetermined price changes, 712 (Customer Class) and 714 (Good Credit). There are a number of different price change operators in accordance with the present invention including, but not limited to, maximum, summation, and minimum. In each instance, the price change operation performs the designated operation on one or more predetermined price changes that are dependent upon the compound price change.

[0074] Some functions 270 in accordance with the present invention display a price trail. In such displays, the price trail ending point is computed based on the plurality of price changes found within the price trail. The display and computation of a price trail is illustrated by the price trail 800 in Fig. 8. In Fig. 8, the staring point of price trail 802 is the list price and the ending point of the price trail is the sales price 820. Illustration of the price trail in the format shown in Fig. 8 is advantageous because it allows the user to see how quote is calculated. Advantageously, those aspects of the price trail that cannot be modified are merely summarized by the display. Furthermore, those aspects of the price trail that the user still has discretion over can be modified by the user. Upon modification, the user can recompute the price trail to see how the modification affects the ending point of the price trail. To illustrate these concepts, consider price adjustment 804, the volume discount adjustment. Since the volume discount has been set up to be a function of the number of units ordered, the user cannot change the absolute value of adjustment 804. Of course, the user can always modify the quote so that a different quantity is ordered. However, once a quantity has been decided upon, the volume discount is fixed. In contrast, price change 806 represents an optional price discount, the key customer discount. Toggle 807 allows the user to activate usage of this discount. If toggle 807 is selected, price change 806 is used in the computation of ending point 804 of price trail 800. Similarly, price change 808 and 810 can be changed by the user. If the user makes a change, price trail 800 is recomputed in order to update the value of ending point 804.

[0075] Some embodiments of the present invention provide a price change operator in a compound price change that allows for user selection between a plurality of predetermined price changes. Price change 808 illustrates one such price change. In addition, some embodiments of the present invention provide a price change operator in a compound price change that provides a user directed choice between accepting or rejecting the application of one or more predetermined price changes. Price change 806 illustrates one such price change. When the user alters a price change in a price trail, the price trail is recomputed based on each of the price changes in the price trail.

[0076] Some embodiments of the present invention provide a special form of price change (*e.g.*, a suggestion). This special form of price change is actually a rationalization or suggestion for why the user should apply one or more predetermined price changes. For example, the special form of price change may offer the user a suggestion on how a better price break could be achieved by modifying the quote. For example, if the standard quantity breaks are 0-100 (-1%), 101-200 (-2%), 201 or more (-3%), and a customer orders 98 widgets, a suggestion could indicate that ordering a few more items would give a larger discount. Another price change in accordance with the present invention is a price change that replaces the value of the starting point of the price trail. Yet other price changes in accordance with the present invention add or subtract a percentage of the starting price value or a fixed amount to the value of the starting point of the price trail. Still other price changes in accordance with the present invention provide a user directed choice to specify a value for the ending point of the price trail. Price change 812 illustrates one such price change.

[0077] Some embodiments of the present invention provide a price change that provides a user directed choice for specifying a discount. Further, the absolute size of the discount that will be accepted is determined by a characteristic of the user. Price change 810 (Fig. 8) illustrates one such price change. In Fig. 8, user 814 has the discretion to introduce a maximal ten percent discount into price change 810. After a value is entered into 814, price trail 800 is recomputed based on each of the price changes in the price trail.

PRICE AGREEMENTS 288

[0078] As referenced above, memory 224 also includes one or more price agreements 288. A price agreement 288 is a data structure that defines how products are priced in the transactions governed by the agreement 288. Fig. 10 illustrates a price agreement 288 in accordance with one embodiment of the present invention. The exemplary price agreement 288 include a price agreement name 1000 and defines one or more payment terms 1002. Optionally, the price agreement specifies the price model 256 to be used as well as one or more agreement-specific price tables 254. A price agreement may apply to many customers. In addition, multiple price agreements may apply to different accounts belonging to a single customer.

[0079] Fig. 4 describes processing steps in accordance with one embodiment of the invention that shows how a price engine 250 and a price agreement 288 work together to form a quote for a product. In step 402, a price engine 250 is activated. The activated price engine 250 is any of the price engines 250 in memory 224 (Fig. 2). Then, in response to a request for a quote from a user, the activated price engine 250 provides a price form 252 that provides the fields necessary to form a quote. Such fields include a product selection field and a product quantity selection field. In some embodiments, the price form 252 furnished in step 404 requires the user to select a price agreement 288 that will be used to govern the transaction. In step 406, the user specifies, using the price form, which products or services in the product catalog 294 are desired as well as desired quantities. In addition, in instances where the identity of the price agreement 288 to be used is not dictated by the price form 252, the user specifies a price agreement 288. The price agreement 288 selected by the user specifies which price model 256 in memory 224 will be used by price engine 250 to govern the transaction. In some instances, a price agreement 288 may specify specific price tables 254 that will be used to govern the transaction. Price tables 254 that are limited to specific agreements are referred to as agreement-specific price tables. It will be appreciated that a price agreement 288 does not have to identify specific price tables 254. In instances where a price agreement 288 does not identify agreement-specific price tables, price tables 254 that serve as global price tables are used to govern the transaction. The price model 256 specified by the selected price agreement 288 determines which waterfall reference structure 260 and which price forms 262 will be used for the transaction. In addition, the selected price agreement 288 determines which price forms 252 will be available to the user to handle various aspects of the transaction, such as viewing how the quote price was computed. In processing step 408, the activated price engine 250 computes a quote using the price model 256 designated by the selected agreement 288 as well as any agreement-specific price tables designated by the agreement 288.

WATERFALL HISTORY DATABASE 290

[0080] Waterfall history database 290 is a novel feature of the present invention that provides the tools necessary to optimize corporate pricing strategy even at the transactional level. The waterfall history database 290 is composed of one or more price trail data structures 291 (Fig. 1). As described above, prices for product quotes are calculated using price functions 270. Price functions 270 create a price trail record (price trail) that is stored in a specified price trail data structure 291. Information recorded by price function 270 includes not only the final price calculated, but also the specific adjustment amounts applied to calculate the particular price. The activated price engine 250 writes each price trail record to the appropriate price trail data structure 291 in the waterfall history database 290. Thus, the waterfall history database 290 tracks not only prices, but also all adjustments that were applied to create each individual price in each transaction performed using system 210 (Fig. 2). In this way, system 210 is capable of supporting corporate price optimization down to the transactional level.

[0081] Fig. 5 illustrates the architecture of pricing trail records 500 in an exemplary price trail data structure 291. Each exemplary record 500 corresponds to a unique transaction that took place using system 210. Each exemplary record 500 includes the name of the product (SKU) 502 that was sold during the transaction, the customer who bought the product (504), the source of the adjustment (504) (*i.e.*, the price form 252 and line item number where the price was calculated), and the transaction date 508. The exemplary record 500 further includes the price point type and/or price adjustment (510). A price point is a node in a waterfall graph that represents an average price for a product at a given stage. Exemplary price points, therefore, include list price (*i.e.*, the price listed in product catalog 294 for a quantity of one item of the product), net price (*i.e.*, the amount of payment received for one item after all on-invoice and off-invoice adjustments are applied against the list price), and invoice price (*i.e.*, the price received before off-invoice adjustments are applied). A price adjustment category represents types of price adjustments that are used to add to or subtract from the value of a price point. Exemplary adjustment categories include, but are not limited to, an order size discount, a promotion discount, an exception discount, a cash discount, a cooperative advertising discount, salesperson discretion, a promotional bonus, a product line rebate, an annual volume bonus, a marketing allowance, and a freight surcharge. Each exemplary record 500 further includes a quantity sold 512, the unit amount sold 514, the total product sold (*i.e.* quantity * unit amount) 516, the lot number from which the product was sold (520) and which price adjustment conditions fired in the price function 270 that corresponds to the price trail data structure 291.

[0082] The following illustrative pseudocode shows how a waterfall trail data structure 291 can be defined in accordance with one embodiment of the present invention:

```
501    waterfall MyCoolStructure {
502      pricepoint "List Price" { }
503      adjustment "Quantity Discount" {
504    description "Discount for buying in bulk";   }

505      adjustment "Standard Discount" {}
506      adjustment "Fixed Discounts" {
507        adjustment "Fixed Discount" {}
508        adjustment "Fixed Price" {}   }

509      adjustment "Promotion" {}
510      adjustment "Key Customer Discount" {}
511      adjustment "Tax" {}
512      adjustment "Freight" {}
513      adjustment "Sales discretion" {}
514      adjustment "Price Limits" {
515        adjustment "Standard Price Limit" {}
516        adjustment "Premium Price Limit" {}   }

517      pricepoint "Invoice Price" { }
518      adjustment "Unit Rebate" {}
519      adjustment "Volume Rebate" {}

520      pricepoint "Net price" {} }
```

Line 501 of the exemplary code defines the price trail definition structure 291 "MyCoolStructure". Further, the code defines three price points for exemplary structure 291. They are "List Price" (line 502), "Invoice Price" (line 517), and "Net price" (line 520). For each of these price points, a number of price adjustments are defined, such as "Quantity Discount" (line 503). The pseudocode further provides a description for some of the pricing adjustments. For example, line 504 of the exemplary pseudocode describes the price adjustment "Quantity Discount" as a "Discount for buying in bulk". A novel feature of the present invention is that price adjustments may have sub-price adjustments. For example, the price adjustment "Fixed Discount" (line 506) has two sub-price adjustments, namely "Fixed Discount" per se (line 507) and "Fixed Price" (line 508). In the illustrative pseudo-code, sub-price adjustments are place within curly brackets after a parent price adjustment.

[0083] Use of the waterfall history database 291 in system 200 is advantageous because it allows for the convenient tracking of the various off-invoice discounts on a customer or transaction basis. Such off-invoice discounts include payment term discounts, cooperative advertising, and customer-specific freight. Tracking of such items provides the foundation for optimizing pricing strategies at the transactional level.

WATERFALL REPORTING MODULE

[0084] Some embodiments of the present invention provide a waterfall reporting module 292 for graphically depicting a plurality of price adjustments that are applied to a product set over a predetermined time period. A product set may consist of a single shopkeeping unit (SKU) that uniquely represents a product in a product catalog. Alternatively, a product set may comprise each product sold to a particular customer, a particular group of customers, a specified list of customers, or virtually any other set of products selected by a user of the system, during the predetermined time period. Groups of customers may be created using a metric such as "customers with a high credit rating" or "high volume customers." The predetermined time period is any period of time, such as a fiscal year, a quarter, the past week, or the past hour.

[0085] In some embodiments in accordance with the present invention, waterfall reporting module 292 includes instructions for receiving a request for a graphical depiction of the plurality of price adjustments that are applied to a product set over the predetermined time period. In response to this request, transaction data for the plurality of price adjustments for the product set during the predetermined time period are accessed by waterfall reporting module 292 from waterfall history database 290. Then, for each price adjustment in the plurality of price adjustments under consideration, a representation of an amount the price of the product set was adjusted in accordance with the corresponding price adjustment during the time period is calculated using the transaction data. The waterfall pricing module also has instructions for graphing each representation as an element in a graph. The graph has a start price point and an end price point. Each calculated element is placed between the start price point and the end price point of the graph.

[0086] The utility and advantages of waterfall reporting module 292 is shown in the following example. Because every price function constructs a price trail and stores the price trail as a record in a price trail data structure 291 in waterfall history database 290, the exemplary transaction data found in Table 4 below may be retrieved from database 290.

Table 4 - Exemplary transaction data stored in database 290

SKU #	CUST	Form #	Entry Type	Quantity	Amount	Total
#123	Acme	I1	ListPrice	100	$100	$10000
#123	Acme	I1	QtyDisc	100	-$10	-$$1000
#123	Acme	I1	CustDisc	100	-$5	-$$500
#123	Acme	I1	Freight	100	$0.98	$98
#123	Acme	I1	Net15	100	-$1.47	-$147
#123	Acme	I2	ListPrice	100	$80	$8000
#123	Acme	I2	QtyDisc	100	-$10	-$1000
#123	Acme	I2	CustDisc	100	-$10	-$1000
#123	Acme	I2	Freight	100	$0.92	$92
#123	Acme	I2	Net15	100	-$0.92	-$92
#123	Acme	R1	SKUIncentive	10	-$1	-$10
#123	Acme	R1	SKUIncentive	50	-$4	-$200
#123	Acme	R2	VolumeIncentive	200	-$.74	-$147

Each row in Table 4 represents a different aspect of a sale of product 123 to the Acme corporation during the first quarter of 2002. In Table 4, column 1 (SKU #) is a unique product identifier. In this case, each entry in table 4 refers to the product having the SKU number 123. Column 2 (CUST) refers to the customer that bought the product. Column 3 (Form #) refers to the price form 252 that was used to purchase the product. In this case there are four different price forms in use, I1, I2, R1, and R2. Column 4 (Entry type) refers to the name of a price adjustment or the name of a price point (start, end, or intermediate) that the row references. Columns 5 through 7 respectively reference quantity sold, unit amount, and total (quantity * unit amount).

[0087] The transaction data shown in Table 4 may be used to create a graph such as graph 900 (Fig. 9). Graph 900 has a start price point (List Price 902) and an end price point (Net Price 906). Between the start and end price point, there is an intermediate price point (Invoice Price 904). Between the price points there are a number of calculated elements. Each element is a representation. Each representation corresponds to an amount the price of a product set was adjusted in accordance with a particular price adjustment during the predetermined time period as detailed in the transaction data (*e.g.* Table 4). For example, element 908 shows the amount of the price adjustment "volume discount" as applied to a particular product set over a predetermined time period. In some embodiments, the amount that is represented by each element in graph 900 is a summation of the amount the price was adjusted by the price adjustment in each transaction for the purchase of the product set during the predetermined time period. This is the case shown in Fig. 9. In other embodiments, the amount that is represented by each element in graph 900 is a weighted average of the amount the price was adjusted by the corresponding price adjustment in each transaction for the purchase of the product set during the predetermined time period. In such embodiments, the weighting factor is the number of products in the product set. For example a transaction that has three units of SKU #123 will have three times the weight of a transaction for one unit of SKU #123 in computing the magnitude of the representation of a price adjustment that was applied in both transactions.

[0088] As Fig. 9 illustrates, there may be intermediate price points (Invoice price 904) and each element in the graph is associated with either an intermediate price point or the final price point. For example, in graph 900, elements 908 through 920 are associated with intermediate price point 904 and elements 930 and 932 are associated with final price point 906. Each element is plotted in the graph before the price point associated with the element.

[0089] Because of the high level of detail found in the transactional data stored in waterfall history database 290, each price adjustment may include a plurality of subcategories. Therefore, when representations are computed based on the data for each price adjustment, representations for each subcategory in a price adjustment that has subcategories may be computed as well. Such representations indicate the amount the price of the product set was adjusted in accordance with the respective subcategories during the predetermined time period using based upon transaction data. This information for each subcategory, in turn, is graphed. Fig. 9 illustrates this concept. Figure 9 shows the breakdown of the volume discount 908 into the subcategories quantity (908-1), quantity 11-100 (908-2), distributor quantity (908-3), and premium quantity (908-4). The use of subcategories in a waterfall graph is highly advantageous because it allows for thorough investigation of the effectiveness of a corporate pricing strategy at a high level of detail. Such detail is only possible by the use of a waterfall history database 290 that tracks every factor used in the computation of every transaction pocket price.

Alternate Embodiments

[0090] The present invention can be implemented as a computer program product that includes a computer program mechanism embedded in a computer readable storage medium. For instance, the computer program product could contain the program modules shown in Fig. 2. These program modules may be stored on a CD-ROM, magnetic disk storage product, or any other computer readable data or program storage product. The software modules in the computer program product may also be distributed electronically, via the Internet or otherwise, by transmission of a computer data signal (in which the software modules are embedded) on a carrier wave.

[0091] The relative location of data structures and program modules in memory 224 of computer 220 as shown in Fig. 2 is merely intended to show the relationship between such structures. One of skill in the art will appreciate that the invention is not limited by the locations of particular data structures and modules in Figure 2. For example, there is no requirement that price tables 252 exists within a price engine 250 construct. In fact, price tables 252, and even price forms 250 and price models 256 may be wholly independent of price engines 250. That is, they may be stored as independent constructs in memory 224. In fact, they may be found in any location that can be referenced by system 200. The same is true for constructs such as price functions 270. Price functions 270 may, for example, be stored as independent constructs in memory 224.

[0092] While the present invention has been described with reference to a few specific embodiments, the description is illustrative of the invention and is not to be construed as limiting the invention. Various modifications may occur to those skilled in the art without departing from the true spirit and scope of the invention as defined by the appended claims.